Universitatea Alexandru Ioan Cuza

Facultatea de Informatica



Proiect: Connect4

Gherman Dan-Gabriel

Anul II, grupa A3

danghermang@gmail.com|dan.gherman@info.uaic.ro

##### Connect4 (B)

Simulati jocul Connect Four intr-o aplicatie client/server. Fiecare client se conecteaza la server unde acesta va stabili jucatorul care face prima mutare si culoare cu care va juca fiecare. Serverul are rolul de a tine scorul, jocul putant fi jucat pe mai multe reprize, precum si rolul de a afisa fiecarui jucator gridul fiecarui jucator, dupa ce se efectueaza o mutare.

1. **Introducere.....................................................2.**
2. **Tehnologiile utilizate......................................7.**
3. **Arhitectura aplicației.....................................8.**
4. **Detalii de implementare.................................9.**
5. **Concluzii........................................................16.**
6. **Bibliografie....................................................16.**

# **Introducere**

Connect Four este un joc pentru cu doi jucători, în care jucătorii aleg mai întâi o culoare și apoi dau drumul la discuri colorate din partea de sus a unei matrici cu șapte coloane si șase rânduri, avand grila suspendată pe verticală. Piesele de cad direct în jos si ocupă următorul spațiu disponibil în interiorul coloanei. Obiectivul jocului este de a fi primul pentru a forma o linie orizontală, verticală sau diagonală a patru discuri de aceiasi culoare. Primul jucător, poate câștiga întotdeauna daca face mutarile potrivite.

# **Tehnologii Utilizate**

TCP:

**Protocolul de control al transmisiei** (sau **TCP**, de la engl. *Transmission Control Protocol*) este un protocol folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o adresă IP și de către un port TCP.

Transmission Control Protocol (TCP) este unul dintre protocoalele de bază ale suitei de protocoale Internet. TCP este unul dintre cele două componente originale ale suitei (celălalt fiind Protocolul Internet, sau IP), astfel încât întreaga suita este frecvent menționată ca stiva TCP/IP. În special, TCP oferă încredere, asigura livrarea ordonata a unui flux de octeți de la un program de pe un computer la alt program de pe un alt computer aflat în rețea. Pe lângă sarcinile sale de gestionare a traficului, TCP controlează mărimea segmentului de date, debitul de informație, rata la care se face schimbul de date, precum și evitarea congestionării traficului de rețea. Printre aplicațiile cele mai uzuale ce utilizează TCP putem enumera World Wide Web (WWW), posta electronica (e-mail) și transferul de fișiere (FTP).

Cu ajutorul protocolului TCP s-a realizat legătura dintre server si clienți, prin socket-urile dintre server si client realizându-se atât transmiterea intenției unui client de a se conecta la un joc de șah, cat si transmiterea matricei ce reprezintă tabla de șah propriu-zisa.

Threads:

In mod obisnuit un proces poate fi considerat ca single threaded - executia sa constand intr-un fir principal, reprezentat de proces in sine. Prin crearea de threaduri, procesul poate executa diferite secvente de cod in paralel pe fiecare din firele de executie. Spre deosebire de procesele fii lansate in cadrul unui proces parinte, threadurile create vor fi parte componenta a procesului care le-a creat, in consecinta partajand spatiul de executie al procesului (variabilele declarate in procesul "parinte" de exemplu sunt comune tuturor threadurilor).   
Asemeni unui proces, si unui thread ii este asociat un ID ce are ca tip de data pthread\_t.

# **Arhitectura aplicației**

Aplicația este construita dintr-un server care are rolul unui manager al meciurilor de connect4, aceasta realizeaza cate un nou thread, reprezentând meciul, doar in momentul in care se conectează o pereche de 2 clienți, cat si rolul de punte de comunicare dintre cei 2 clienți, transmiterea unei mutări intre clientul I si J se realizează prin intermediul protocolului TCP implementat in server. (folosind primitivele read si write, scriind prin intermediul socket-ului ce realizeaza legatura intre client si server)

Clientul are o interfata realizata in consola, reprezentarea tablei de joc fiind facuta in ASCII art, iar verificarea validității mutărilor va fi realizata local tot in acesta, pentru a evita transmiterea de informație inutila intre server si client. Conditiile de joc sunt verificate insa in server, iar in momentul terminarii jocului serverul transmite mesajul de inchidere catre clientii partidei respective.

De asemenea, serverul are si o tabela in care tine evidenta castigurilor.

Clienteul are integrat si un mod singleplayer, in care alege o culoare si poate juca de unul singur, miscarile oponentului fiind facute de catre masina.

Primul client din pereche care se conectează la server va juca cu piesele galbene, si va face prima mutare.

In momentul in care un client verifica conditia de castig sau de draw acesta va afișa „mesajul de victorie” si va transmite rezultatul si celuilalt client prin intermediul server-ului.

Partidele sunt concurente, putandu-se juca mai multe in acelasi timp fara a avea interferente in timpul mutarilor.

Tipul de conexiune dintre server si clienți poate fi reprezentat prin diagrama următoare:

# **Detalii de implementare**

**Pentru server folosim:**

/\*structura de date in care encapsulam id-ul, ambii jucatori si board-ul pentru fiecare joc de connect4 pornit.\*/

typedef struct thData{

int idThread;

int player\_yellow;

int player\_red;

int board[10][10];

int array[10];

}thData;

/\* crearea unui socket \*/

if ((sd = socket (AF\_INET, SOCK\_STREAM, 0)) == -1)

{

perror ("[server]Eroare la socket().\n");

return errno;

}

/\* utilizarea optiunii SO\_REUSEADDR \*/

int on=1;

setsockopt(sd,SOL\_SOCKET,SO\_REUSEADDR,&on,sizeof(on));

/\* pregatirea structurilor de date \*/

bzero (&server, sizeof (server));

bzero (&from, sizeof (from));

/\* umplem structura folosita de server \*/

/\* stabilirea familiei de socket-uri \*/

server.sin\_family = AF\_INET;

/\* acceptam orice adresa \*/

server.sin\_addr.s\_addr = htonl (INADDR\_ANY);

/\* utilizam un port utilizator \*/

server.sin\_port = htons (PORT);

/\* atasam socketul \*/

if (bind (sd, (struct sockaddr \*) &server, sizeof (struct sockaddr)) == -1)

{

perror ("[server]Eroare la bind().\n");

return errno;

}

/\* punem serverul sa asculte daca vin clienti sa se conecteze \*/

if (listen (sd, 2) == -1){

perror ("[server]Eroare la listen().\n");

return errno;

}

/\* functia executata de fiecare thread ce realizeaza comunicarea cu clientii \*/

static void \*prepare\_match(void \* arg){

struct thData tdL;

tdL= \*((struct thData\*)arg);

printf ("[thread]- %d - Asteptam mesajul...\n", tdL.idThread);

fflush (stdout);

start\_game((struct thData\*)arg);

return(NULL);

};

/\*primire de mesaj de la client\*/

client\_read = read(tdL.player\_yellow, mesaj1, 50);

mutare=atoi(mesaj1);

printf("\nprimit %d\n",mutare);

if (client\_read < 0){

printf("[Thread %d]\n",tdL.idThread);

perror ("Eroare la read() de la client.\n");

}else if(client\_read == 0){

shutdown(tdL.player\_yellow, 2);

mutare=WIN\_RED;

castig\_rosu++;

char mesaj[100];

snprintf(mesaj,50,"%d",mutare);

if(write (tdL.player\_red, mesaj, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

shutdown(tdL.player\_red, 2);

printf("Partida %i terminata!\n", tdL.idThread);

showStats(castig\_rosu,castig\_galben);

break;

}

\*/verificare conditii sfarsit joc\*/

if(win){

char mesaj[100];

mutare=WIN\_YELLOW;

castig\_galben++;

snprintf(mesaj,50,"%d",mutare);

if(write (tdL.player\_yellow, mesaj, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

shutdown(tdL.player\_yellow, 2);

if(write (tdL.player\_red, mesaj, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

shutdown(tdL.player\_red, 2);

printf("Partida %i terminata!\n", tdL.idThread);

showStats(castig\_rosu,castig\_galben);

break;

}

if(end){

char mesaj[100];

mutare=DRAW;

snprintf(mesaj,50,"%d",mutare);

if(write (tdL.player\_yellow, mesaj, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

shutdown(tdL.player\_yellow, 2);

if(write (tdL.player\_red, mesaj, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

shutdown(tdL.player\_red, 2);

printf("Partida %i terminata!\n", tdL.idThread);

showStats(castig\_rosu,castig\_galben);

break;

}

/\*trimitere mesaj la client\*/

if (write (tdL.player\_yellow, mesaj4, 50) <= 0){

printf("[Thread %d] ",tdL.idThread);

perror ("[Thread]Eroare la write() catre client.\n");

}

**Pentru client folosim:**

/\*generator de numere random\*/

int randint(int n) {

if ((n - 1) == RAND\_MAX) {

return rand();

} else {

long end = RAND\_MAX / n;

assert (end > 0L);

end \*= n;

int r;

while ((r = rand()) >= end);

return r % n;

}

}

/\*afisare a tabelei de joc folosind ascii art\*/

void afisare(int board[10][10])

{

system("clear");

int i,j;

PRINT\_BLUE("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

NEWLINE;

for(i=5;i>=0;i--)

{

for(int it=0;it<2;it++)

{

PRINT\_BLUE("■■");

for(j=0;j<7;j++)

{

switch(board[i][j])

{

case GALBEN:

PRINT\_YELLOW("■■");

break;

case LIBER:

printf(" ");

break;

case ROSU:

PRINT\_RED("■■");

break;

}

PRINT\_BLUE("■■");

}

NEWLINE;

}

PRINT\_BLUE("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

NEWLINE;

}

PRINT\_BLUE("■■ 1 ■■ 2 ■■ 3 ■■ 4 ■■ 5 ■■ 6 ■■ 7 ■■");

NEWLINE;

}

/\*functii de verificare a validitatii mutarilor\*/

int move(int board[10][10],int array[10],int mutare,int color)

{

int index=array[mutare];

if(array[mutare]>=6)

return 0;

else

{

board[index][mutare]=color;

array[mutare]++;

return 1;

}

}

int check\_win(int board[10][10],int height,int width,int color)

{

int x,y,check;

//horizontal

check=1;

y=width-1;

while(board[height][y]==color&&y>=0)

{

check++;

y--;

}

y=width+1;

while(board[height][y]==color&&y<=6)

{

check++;

y++;

}

if(check>=4)

{

return 1;

}

else{

//vertical

check=1;

x=height-1;

while(board[x][width]==color&&x>=0)

{

check++;

x--;

}

x=height+1;

while(board[x][width]==color&&x<=5)

{

check++;

x++;

}

if(check>=4)

{

return 1;

}

else{

//diagonal 1

check=1;

x=height-1;

y=width-1;

while(board[x][y] == color && x>=0 && y>=0 )

{

check++;

x--;

y--;

}

x=height+1;

y=width+1;

while(board[x][y] == color && x<=5 && x<=6 )

{

check++;

x++;

y++;

}

if(check>=4)

{

return 1;

}

else{

//diagonal 2

check=1;

x=height-1;

y=width+1;

while(board[x][y] == color && x>=0 && y<=6 )

{

check++;

x--;

y++;

}

x=height+1;

y=width-1;

while(board[x][y] == color && x<=5 && y>=0 )

{

check++;

x++;

y--;

}

if(check>=4)

{

return 1;

}

else{

return 0;

}

}

}

}

}

int check\_end(int array[10])

{

for(int i=0;i<7;i++)

{

if(array[i]<6)

{return 0;}

}

return 1;

}

# Concluzii

Pentru acest proiect se folosește un protocol TCP pentru a realiza legătura dintre doi jucători de connect4, cat si o interfață grafica simplista, realizata in ASCII art.

Pe viitor as dori sa implementez o componenta de versus ai pe nivele diferite de dificultate, o tabela de „High score” care sa păstreze un clasament in funcție de cea mai scurta durata pentru a ajunge la finalul jocului, sau cea cele mai puține mutări realizate, cat si posibilitatea de a juca pe mobil.

# Bibliografie

* <https://ro.wikipedia.org/wiki/Qt>
* <http://profs.info.uaic.ro/~adria/teach/courses/net/cursullaboratorul.php>
* <http://profs.info.uaic.ro/~eonica/rc/>
* <http://stackoverflow.com>
* <https://ro.wikipedia.org/wiki/Protocol_de_control_al_transmisiei>
* [www.springer.com](http://www.springer.com)
* https://github.com/latextemplates/LCNS